Math 142: Modeling **Name:**_____

Homework # 4 **Due: Friday Oct 27**

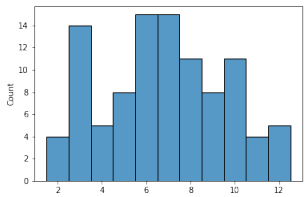
Directions Complete the exercises. Your solutions to the exercises should be submitted to Gradescope before the indicated due date above. Please follow rules regarding Gradescope submission as described in the syllabus.

References Except for the help of the instructor or TAs and the class textbooks and notes, if you use any resources, for example, a book, a website, or you discussed with your friends, please acknowledge them in this References section.

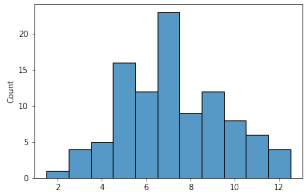
- I discussed Problem ?? with STUDENT A, STUDENT B, ...
- I used BOOK/WEBSITE to help me do Problem ??.

Exercises

1. (a) Create a model of a pair of dice analogous to the coin model where there is an equal probability for each of the 6 numbers on each die. Run the model 100 times and make a graph of the resulting distribution. (Do NOT google for this code. Just tweak the coin model.) Plot the results.



   (b) Create a second version where the dice are biased so that there is a 30% greater chance of a six on each die. Plot the results.



   (c) Compare the two graphs and describe the results.

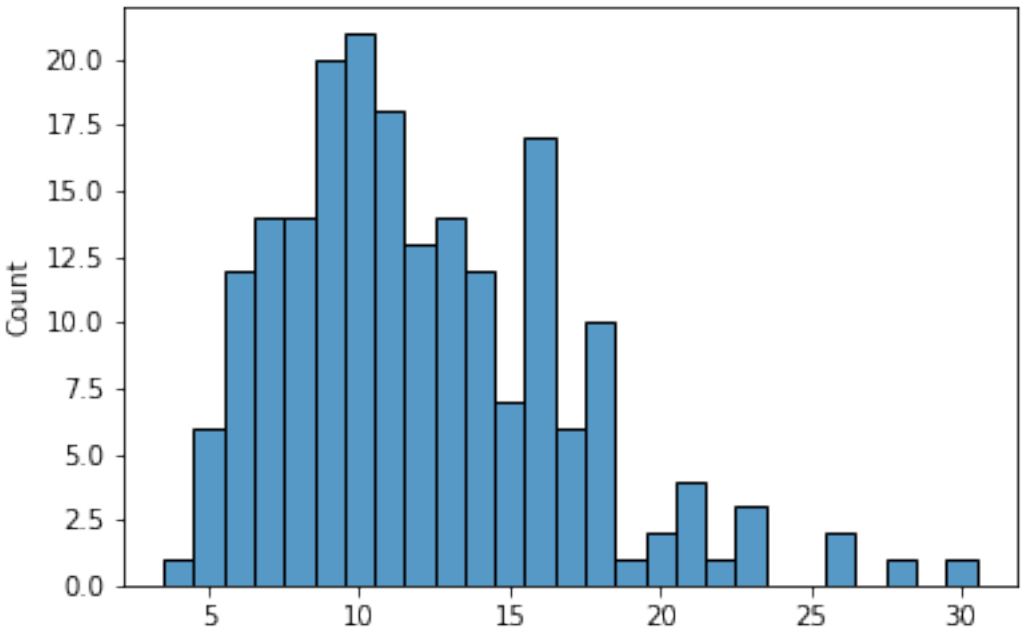
   The top graph is more symmetric than the bottom graph because of a higher probability of a 6s getting rolled in the bottom graph. The lower graph is slightly skewed to the left.

2. **Length of a polymer** Versions of the random walk model we created in class are used in polymer physics to model the sizes of polymers, such as strands of RNA. We imagine the polymer as a set of links, each of length 1. These links are arranged back and forth along a straight line (so the first link goes toward the left or right with equal probability, similarly the second, and we can simulate the

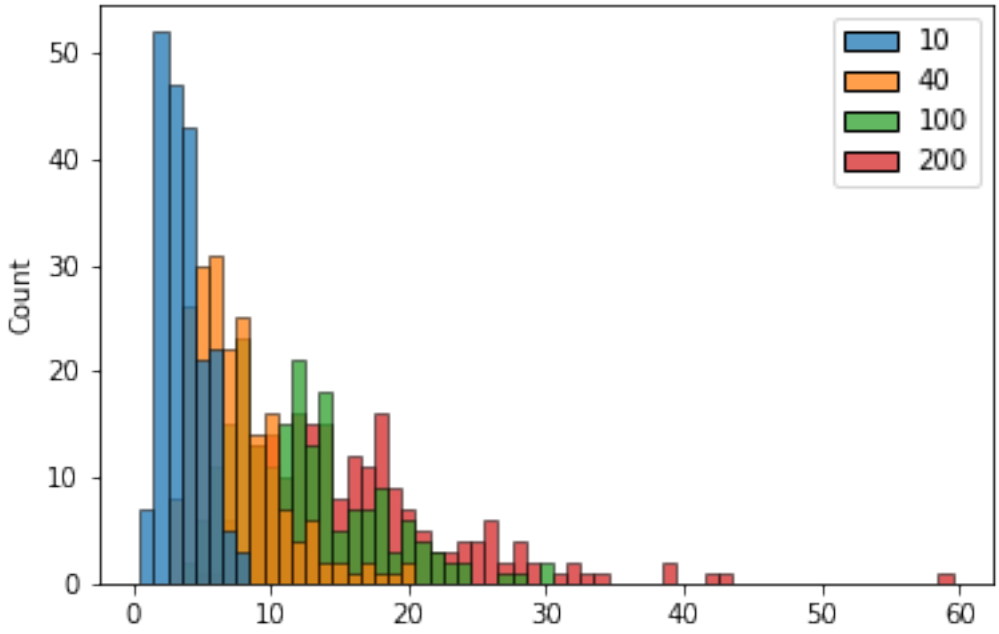
configuration of a polymer with N links as a N-step random walk).

A quantity of interest is how the maximum distance polymer gets from its starting position (0) if it has N links. This distance is referred to as the radius of gyration of the polymer.

(a) Make a histogram of the observed radius of gyration of 100 polymers, each of which have 200 links. Notice that the histogram is not bell shaped.



(b) It has been argued that the average radius of gyration is proportional to $\sqrt{N}$. By running 100 replicate simulations each with polymers of lengths $N = 10, 40, 100, 200$, and measuring the average radius of gyration for each length of polymer, test this theory.



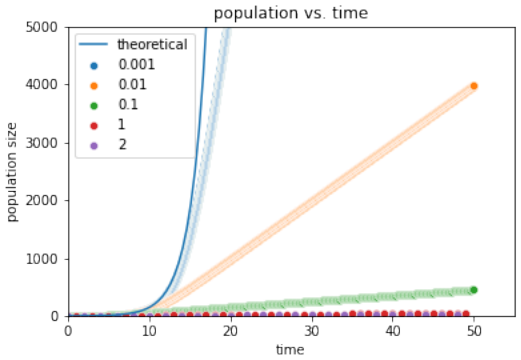
The squares of our average lengths were $12.96, 55.6516, 151.5361$, and 278.723025 respectively. While these are somewhat similar to the number of lengths, $\sqrt{N}$ is probably an underestimate of the average length.

Real polymers are arranged in 3D, but our model can describe how they may behave when confined (e.g. in virus).

3. In class we developed a mathematical model for stochastic population growth. Throughout this question, unless a question part instructs you to do otherwise, you may use the same parameters we adopted in class: $b = 0.5$, $\Delta t = 0.01$.

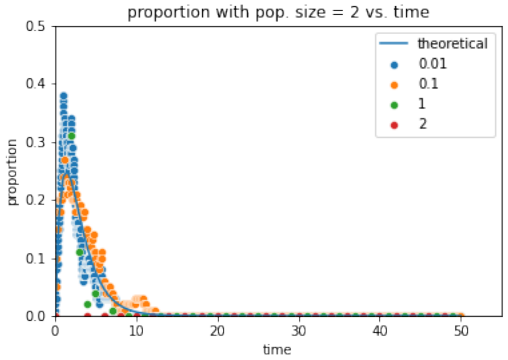
   (a) Briefly describe in your own words, how we simulate numerically stochastic population growth.
   Our goal is to use randomness to predict the population size at a time step t. Between each time step, every individual has the chance to reproduce (or die). We use a random number generator to determine whether an individual is born/dies between censuses. We choose a time step small enough s.t the probability that the population changes by more than 1 between time steps is insignificant. Repeating the experiment several times, we can average the results from each experiment to determine the average population growth.

   (b) In class we will explain why it is necessary to assume that Δt is very small. Let's see why in your simulations. Plot on the same axes, graphs of the mean population size against time, among your 100 replicate simulations, when $\Delta t = 0.01, \Delta t = 0.1, \Delta t = 1, \Delta t = 2$. On the same plot draw the average population size that we predict theoretically: $N(t) = e^{bt}$. Comment on whether the simulations still follow theory for larger values of Δt.



(c) From the 100 replicate simulations, make a plot of the fraction of the simulations that have exactly two cells, as a function of time, t. On the same plot include the theoretical curve (which we will derive in class):

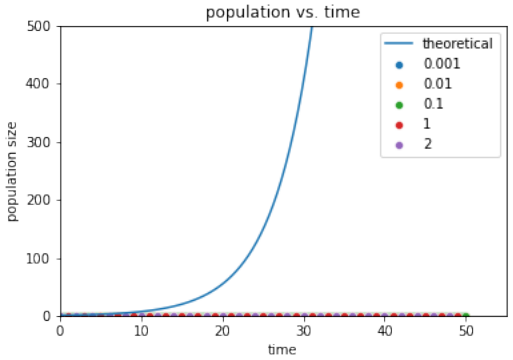
$$P_2(t) = e^{-bt}\left(1 - e^{-bt}\right).$$



(d) In our simulation we have neglected the effect of cell deaths. Now let's include this effect. In the code, after you have calculated the number of births (cells to add), consider the possibility that each cell that was present at time t dies with probability $m\Delta t$. Assuming $m = 0.2$, make a plot of the average population size as a function of time t, and compare with the deterministic result:

$$N(t) = e^{(b-m)t}.$$

Also answer: does every single one of your simulated populations grow exponentially?



No. At low population levels ≈ 1 there is a high enough probability that the population could die entirely between the first censuses.

4. We will build a stochastic model for how a population of bacteria is depleted by an antibiotic. Assume that at time $t = 0$, there are precisely N bacterial in the population. Once the antibiotic is added the bacteria stop dividing (that is $b = 0$), and start to die with a mortality rate m. That is, in time Δt, the probability of a given cell dying is $m\Delta t$. We want to calculate the probability distribution, $P_n(t)$, which represents the probability that there are n cells at time t.

   (a) Explain what the initial conditions on $P_n(t)$ are.

   $$P_n(0) = \begin{cases} 1 & \text{for } n = N \\ 0 & \text{for } n \neq N \end{cases}$$

   (b) Show that

   $$\frac{dP_N}{dt} = -NmP_N$$

   and that:

   $$\frac{dP_n}{dt} = (n+1)mP_{n+1} - nmP_n$$

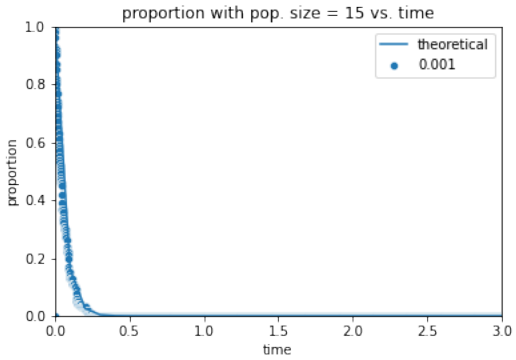
   for $n = 0, 1, 2, \ldots N-1$.
   Notice that because $b = 0$, the population is at most N.
   $P_n(t+\Delta t) = \sum_{k=n}^{N} \binom{k}{k-n} (m\Delta t)^{k-n}(1-m\Delta t)^n P_k(t)$

   $$\frac{P_n(t+\Delta t) - P_n(t)}{\Delta t} = \frac{\sum_{k=n}^{N} \binom{k}{k-n} (m\Delta t)^{k-n}(1-m\Delta t)^n P_k(t)}{\Delta t} - \frac{P_n(t)}{\Delta t}$$

   $\frac{dP_n(t)}{dt} = \lim_{\Delta t \to 0} \frac{P_n(t+\Delta t) - P_n(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{((1-m\Delta t)^n - 1)P_n(t)}{\Delta t} + \frac{(n+1)m\Delta t((1-m\Delta t)^n)P_{n+1}(t)}{\Delta t} + O(\Delta t)$
   $= \lim_{\Delta t \to 0} \frac{(1-nm\Delta t - 1)P_n(t) + O(\Delta t)}{\Delta t} + \frac{(n+1)m\Delta t P_{n+1}(t) + O(\Delta t)}{\Delta t}$
   $= (n+1)mP_{n+1}(t) - nmP_n(t)$ where $P_{N+1}(t) = 0$ for all t.

   (c) Solve the first few differential equation to calculate $P_N(t), P_{N-1}(t)$ and $P_{N-2}(t)$. Can you see any pattern that would help you to guess $P_n(t)$ (reading Chapter 36 in the book might help you with this). You do not need to prove that your answer is correct.
   $P_N(t) = e^{-mNt}$
   $\frac{dP_{N-1}}{dt} + (N-1)mP_{N-1}(t) = Nme^{-Nmt} \Rightarrow (e^{(N-1)t}P_{N-1}(t))' = Nme^{-mt}$
   $\Rightarrow e^{(N-1)t}P_{N-1}(t) = -Ne^{-mt} + c$ Applying initial condition $P_{N-1}(0) = 0$ we obtain $P_{N-1}(t) = Ne^{-(N-1)mt}(1-e^{-mt})$
   $\frac{dP_{N-2}}{dt} + (N-2)mP_{N-2}(t) = N(N-1)me^{-(N-1)mt}(1-e^{-mt})$
   $\Rightarrow (e^{(N-2)t}P_{N-2}(t))' = N(N-1)me^{-mt}(1-e^{-mt})$
   $\Rightarrow e^{(N-2)t}P_{N-2}(t) = \frac{N(N-1)}{2}(1-e^{-mt})^2 + c$ Applying initial condition $P_{N-2}(0) = 0$ we obtain $P_{N-2}(t) = \frac{N(N-1)}{2}e^{-(N-2)mt}(1-e^{-mt})^2$
   $P_n(t) = \binom{N}{N-n} e^{-nmt}(1-e^{-mt})^{N-n}$

   (d) Verify that your solution for $P_N(t)$ is correct, by comparing it with stochastic simulations. Specifically, implement a stochastic simulation for the death of bacteria (you are free to decide for yourself how many replicates you want to run). For sake of definiteness, run the simulations with $0 < t < 10$, and $m = 1.2$, and $N = 15$.

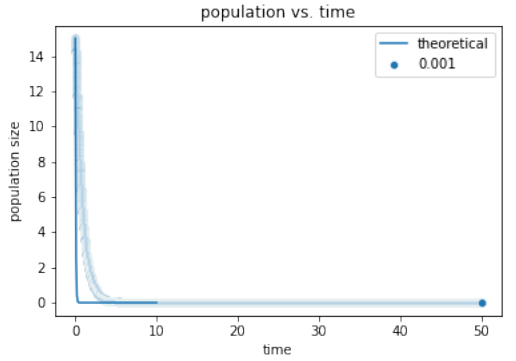


(e) It is hard to see how, on average, the population will change over time, using the formulas for the $P_n(t)$. But we can get this average directly. Suppose that at time t, the average number of cells in the population is $\bar{n}(t)$. Between time t and $t + \Delta t$, how many cells, on average, will die? Use this information to derive a differential equation:

$$\frac{d\bar{n}}{dt} = -m\bar{n}.$$

$\bar{n}(t) = \sum_{n=0}^{N} nP_n(t) = Ne^{-mt}$ by average of binomial dist. $\lim_{\Delta t \to 0} \frac{Ne^{-m(t+\Delta t)} - Ne^{-mt}}{\Delta t} = -Nme^{-mt} \Rightarrow \frac{d\bar{n}(t)}{dt} = -m\bar{n}(t)$

(f) Solve the differential equation from part (e) and show that it agrees with the average population size according to stochastic numerical simulations. Use the same parameter values as in part (d).



5. We are building a model for how a material like concrete degrades due to cracks forming. Let's assume that in a time interval Δt there is a probability $f\Delta t$ of a new crack forming (we call f the fracture rate). Assume moreover that at time $t = 0$ there are no cracks in the material. Let $P_n(t)$ be the probability that there are exactly n cracks in the material at time t.

(a) Show that:

$$\frac{dP_n}{dt} = fP_{n-1} - fP_n \quad , \quad \text{if } n > 0$$

and derive the differential equation for $P_0(t)$. what are the initial conditions for all of your differential equations? $P_n(t) = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases}$

$P_n(t + \Delta t) = f\Delta t P_{n-1}(t) + (1 - f\Delta t)P_n(t) \Rightarrow \frac{P_n(t+\Delta t) - P_n(t)}{\Delta t} = fP_{n-1}(t) - fP_n(t)$. Taking the limit as Δt goes to 0 we obtain $\frac{dP_n}{dt} = fP_{n-1} - fP_n$, and because $P_{-1}(t) = 0$ for all t, $\frac{dP_0}{dt} = -fP_0$

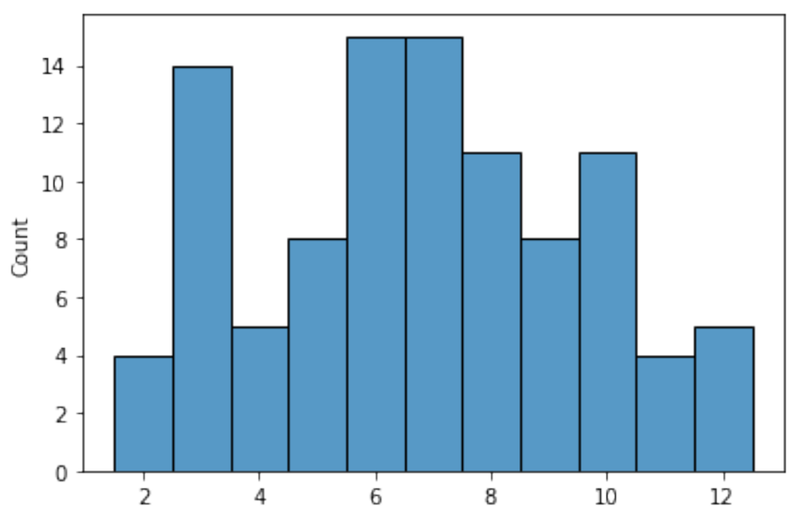
(b) Find a differential equation describing the average number of cracks in each replicate sample. Let $\bar{n}(t)$ be the average number of cracks at time t. $\bar{n}(t+\Delta t) - \bar{n}(t) = (1 + f\Delta t)\bar{n}(t) - \bar{n}(t) \Rightarrow \frac{d\bar{n}(t)}{dt} = f \Rightarrow \bar{n}(t) = e^{ft} - 1$

(c) We have to replace the concrete when it reaches 10 cracks. Assume that $f = 1$. Using stochastic simulations, make a histogram based on 100 replicate simulations, representing 100 samples of the concrete, showing the range of times at which each sample had to be replaced (i.e. reached 10 cracks).

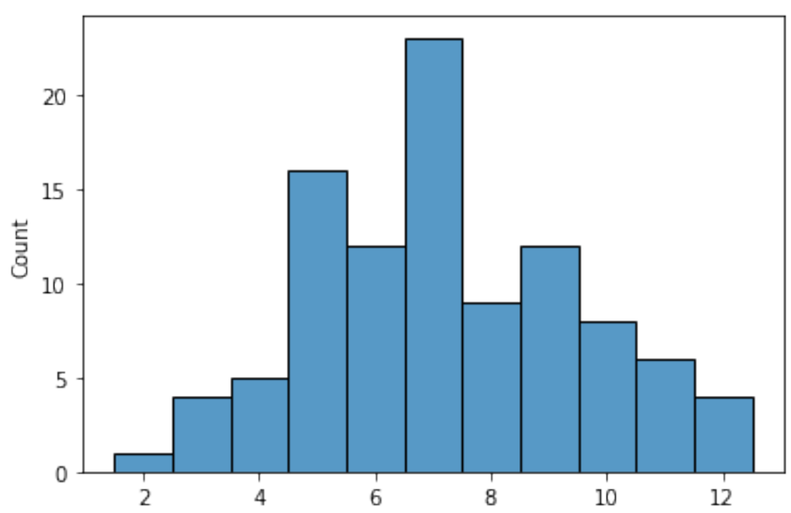
6. Submit the code you used for any and all of the problems. (Print pdf the code) Either lump it all together at the end or when matching problems on Gradescope, select all pages of pdf that has code if you included code within the solution to each answer.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import math
import random
import matplotlib.pyplot as plt
```

In [2]:
```
d_1=np.zeros(100)
randints_1=np.zeros([100,2])
for i in np.arange(100):
    r_1=random.uniform(0,1)
    r_2=random.uniform(0,1)
    d_1[i]=min([math.floor(6*r_1+1),6])+min([math.floor(6*r_2+1),6])
    randints_1[i]=[r_1,r_2]
sns.histplot(data=d_1,discrete=True)
plt.savefig("die_unweighted")
```



In [3]:
```
d_2=np.zeros(100)
randints_2=np.zeros([100,2])
for i in np.arange(100):
    r_1=random.uniform(0,1)
    r_2=random.uniform(0,1)
    randints_2[i]=[r_1,r_2]
    d_2[i]=min([math.floor(63/10*r_1+1),6])+min([math.floor(63/10*r_2+1),6])
sns.histplot(data=d_2,discrete=True)
plt.savefig("die_weighted")
```

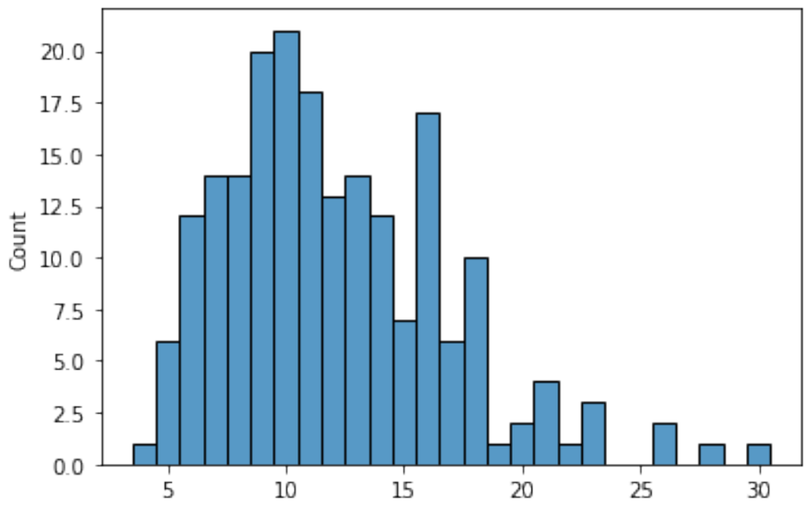


In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import math
import random
import matplotlib.pyplot as plt
```

In [2]:
```
def polymers(simulations,N):
    max_radii = np.zeros(simulations)
    for i in np.arange(simulations):
        x_0=0
        r=0
        for j in np.arange(N):
            v=random.uniform(0,1)
            if v<1/2:
                x_0+=1
            else:
                x_0-=1
            r=max([r,abs(x_0)])
        max_radii[i]=r
    sns.histplot(data=max_radii,discrete=True)
    plt.savefig("100 polymers")
```

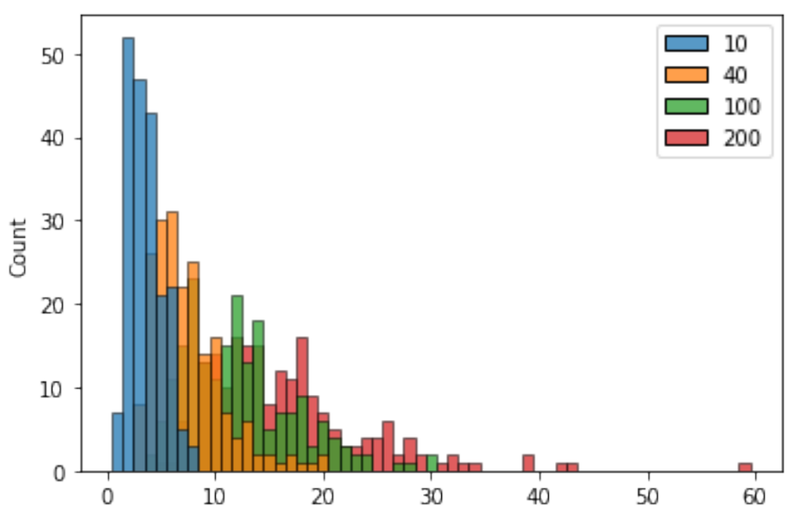
In [3]:
```
polymers(200,100)
```



In [4]:
```
def polymers_2(simulations,N_1,N_2,N_3,N_4):
    max_radii = pd.DataFrame({str(N_1):[],
                              str(N_2):[],
                              str(N_3):[],
                              str(N_4):[]})
    for i in np.arange(simulations):
        x_1=0
        r_1=0
        for j_1 in np.arange(N_1):
            v=random.uniform(0,1)
            if v<1/2:
                x_1+=1
            else:
                x_1-=1
            r_1=max([r_1,abs(x_1)])
        x_2=0
        r_2=0
        for j_2 in np.arange(N_2):
            v=random.uniform(0,1)
            if v<1/2:
                x_2+=1
            else:
                x_2-=1
            r_2=max([r_2,abs(x_2)])
        x_3=0
        r_3=0
        for j_3 in np.arange(N_3):
            v=random.uniform(0,1)
            if v<1/2:
                x_3+=1
            else:
                x_3-=1
            r_3=max([r_3,abs(x_3)])
        x_4=0
        r_4=0
        for j_4 in np.arange(N_4):
            v=random.uniform(0,1)
            if v<1/2:
                x_4+=1
            else:
                x_4-=1
            r_4=max([r_4,abs(x_4)])
        max_radii=max_radii.append(
                          {str(N_1):r_1,
                           str(N_2):r_2,
                           str(N_3):r_3,
                           str(N_4):r_4},ignore_index=True)
    sns.histplot(data=max_radii,discrete=True)
    plt.savefig("10-200 polymers")
    return np.array([np.mean(max_radii[str(N_1)]),np.mean(max_radii[str(N_2)]),np.mean(max_radii[str(N_3)]),np.mean(max_radii[str(N_4)])])
```

In [5]:
```
poly=polymers_2(200,10,40,100,200)
poly**2
```

Out[5]:
```
array([ 12.96    ,  55.6516  , 151.5361  , 278.723025])
```

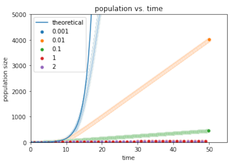


In [ ]:
```
```

In [11]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import random
import math
```

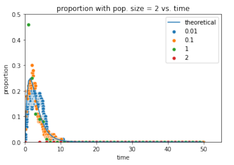
In [33]:
```
def stochastic_sim(step,b,sim_number):
    growth_sim=np.zeros((math.floor(50/step),sim_number))
    growth_average=np.zeros(math.floor(50/step))
    growth_sim[0]=np.ones(100)
    growth_average[0]=1
    for i in np.arange(math.floor(50/step)-1):
        for j in np.arange(sim_number):
            r=random.uniform(0,1)
            if r<b*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]+1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
        growth_average[i+1]=np.mean(growth_sim[i+1])
    sns.scatterplot(y=growth_average,x=np.arange(math.floor(50/step))*step,label=str(step))
```

In [3]:
```
stochastic_sim(0.001,0.5,100)
stochastic_sim(0.01,0.5,100)
stochastic_sim(0.1,0.5,100)
stochastic_sim(1,0.5,100)
stochastic_sim(2,0.5,100)
sns.lineplot(x=np.linspace(0,50,100),y=np.exp(0.5*np.linspace(0,50,100)),label='theoretical')
plt.xlim(0,55)
plt.ylim(0,5e3)
plt.xlabel('time')
plt.ylabel('population size')
plt.title('population vs. time')
plt.savefig('stochastic_q_3_b')
```



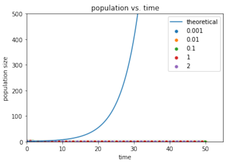
In [32]:
```
def stochastic_sim_2(step,b,sim_number):
    growth_sim=np.zeros((math.floor(50/step),sim_number))
    growth_2=np.zeros(math.floor(50/step))
    growth_sim[0]=np.ones(100)
    growth_2[0]=0
    for i in np.arange(math.floor(50/step)-1):
        for j in np.arange(sim_number):
            r=random.uniform(0,1)
            if r<b*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]+1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
        growth_2[i+1]=(growth_sim[i+1]==2).sum()/sim_number
    sns.scatterplot(y=growth_2,x=np.arange(math.floor(50/step))*step,label=str(step))
```

In [5]:
```
stochastic_sim_2(0.01,0.5,100)
stochastic_sim_2(0.1,0.5,100)
stochastic_sim_2(1,0.5,100)
stochastic_sim_2(2,0.5,100)
sns.lineplot(x=np.linspace(0,50,100),y=np.exp(-0.5*np.linspace(0,50,100))*(1-np.exp(-0.5*np.linspace(0,50,100))),label='theoretical')
plt.xlim(0,55)
plt.ylim(0,0.5)
plt.xlabel('time')
plt.ylabel('proportion')
plt.title('proportion with pop. size = 2 vs. time')
plt.savefig('stochastic_q_3_c')
```



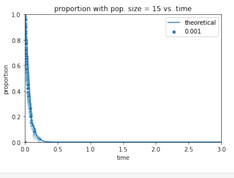
In [46]:
```
def stochastic_sim_3(step,b,m,sim_number):
    growth_sim=np.zeros((math.floor(50/step),sim_number))
    growth_average=np.zeros(math.floor(50/step))
    growth_sim[0]=np.ones(100)
    growth_average[0]=1
    for i in np.arange(math.floor(50/step)-1):
        for j in np.arange(sim_number):
            r_1=random.uniform(0,1)
            r_2=random.uniform(0,1)
            if r_1<b*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]+1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
            if r_2<m*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]-1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
        growth_average[i+1]=np.mean(growth_sim[i+1])
    sns.scatterplot(y=growth_average,x=np.arange(math.floor(50/step))*step,label=str(step))
```

In [7]:
```
stochastic_sim_3(0.001,0.5,0.3,100)
stochastic_sim_3(0.01,0.5,0.3,100)
stochastic_sim_3(0.1,0.5,0.3,100)
stochastic_sim_3(1,0.5,0.3,100)
stochastic_sim_3(2,0.5,0.3,100)
sns.lineplot(x=np.linspace(0,50,100),y=np.exp(0.2*np.linspace(0,50,100)),label='theoretical')
plt.xlim(0,55)
plt.ylim(0,5e2)
plt.xlabel('time')
plt.ylabel('population size')
plt.title('population vs. time')
plt.savefig('stochastic_q_3_d')
```



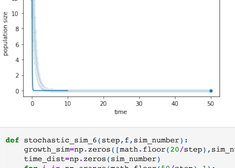
In [38]:
```
def stochastic_sim_4(N,step,m,sim_number,time):
    growth_sim=np.zeros((math.floor(time/step),sim_number))
    growth_2=np.zeros(math.floor(time/step))
    growth_sim[0]=np.ones(sim_number)*N
    growth_2[0]=0
    for i in np.arange(math.floor(time/step)-1):
        for j in np.arange(sim_number):
            r=random.uniform(0,1)
            if r<m*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]-1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
        growth_2[i+1]=(growth_sim[i+1]==N).sum()/sim_number
    sns.scatterplot(y=growth_2,x=np.arange(math.floor(time/step))*step,label=str(step))
```

In [44]:
```
sns.lineplot(x=np.linspace(0,10,100),y=np.exp(-1.2*15*np.linspace(0,10,100)),label='theoretical')
stochastic_sim_4(15,0.001,1.2,100,10)
plt.xlim(0,3)
plt.ylim(0,1)
plt.xlabel('time')
plt.ylabel('proportion')
plt.title('proportion with pop. size = 15 vs. time')
plt.savefig('stochastic_q_4_d')
```



In [56]:
```
def stochastic_sim_5(N,step,m,sim_number):
    growth_sim=np.zeros((math.floor(50/step),sim_number))
    growth_average=np.zeros(math.floor(50/step))
    growth_sim[0]=np.ones(100)*N
    growth_average[0]=N
    for i in np.arange(math.floor(50/step)-1):
        for j in np.arange(sim_number):
            r=random.uniform(0,1)
            if r<m*step*growth_sim[i][j]:
                growth_sim[i+1][j]=growth_sim[i][j]-1
            else:
                growth_sim[i+1][j]=growth_sim[i][j]
        growth_average[i+1]=np.mean(growth_sim[i+1])
    sns.scatterplot(y=growth_average,x=np.arange(math.floor(50/step))*step,label=str(step))
```

In [58]:
```
sns.lineplot(x=np.linspace(0,10,100),y=15*np.exp(-1.2*15*np.linspace(0,10,100)),label='theoretical')
stochastic_sim_5(15,0.001,1.2,100)
plt.xlabel('time')
plt.ylabel('population size')
plt.title('population vs. time')
plt.savefig('stochastic_q_4_f')
```



In [73]:
```
def stochastic_sim_6(step,f,sim_number):
    growth_sim=np.zeros((math.floor(20/step),sim_number))
    time_dist=np.zeros(sim_number)
    for i in np.arange(math.floor(50/step)-1):
        for j in np.arange(sim_number):
            if growth_sim[i][j]<10:
                r=random.uniform(0,1)
                if f*step<r:
                    growth_sim[i+1][j]=growth_sim[i][j]+1
                else:
                    growth_sim[i+1][j]=growth_sim[i][j]
        return np.sum(growth_sim.transpose()!=0)
```

In [74]:
```
stochastic_sim_6(0.001,1,100)
```

Out[74]: 100

In [ ]: